POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

Guide TP2

INF8808E | SUMMER 2022

Version JavaScript

Plan

- More details on labs
- TP2 presentation
- Submission guidelines

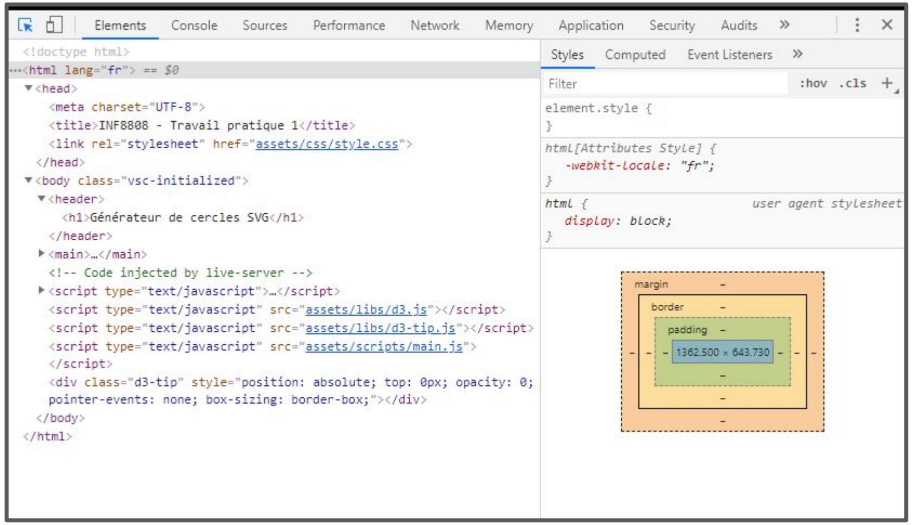
Getting started

Debugging with Chrome

You can also use **Firefox** with very similar steps.

1. Right-click anywhere on the page and select “Inspect” OR Ctrl+Shift+I
2. The inspector will open, which is useful for debugging
3. The “Elements”, “Console”, and “Sources” tabs will be the most useful for these TPs (see next slides)

Chrome inspector



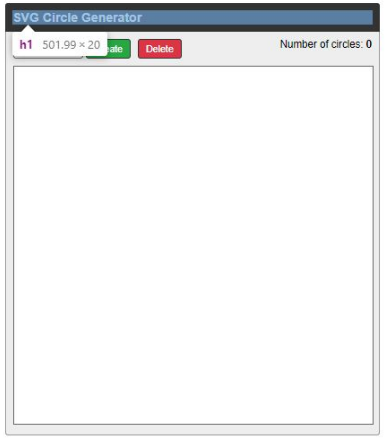
```
<!doctype html>
…<html lang="fr"> == $0
  ▼<head>
      <meta charset="UTF-8">
      <title>INF8808 - Travail pratique 1</title>
      <link rel="stylesheet" href="assets/css/style.css">
    </head>
  ▼<body class="vsc-initialized">
    ▼<header>
        <h1>Générateur de cercles SVG</h1>
      </header>
    ▶<main>…</main>
      <!-- Code injected by live-server -->
    ▶<script type="text/javascript">…</script>
      <script type="text/javascript" src="assets/libs/d3.js"></script>
      <script type="text/javascript" src="assets/libs/d3-tip.js"></script>
      <script type="text/javascript" src="assets/scripts/main.js">
      </script>
      <div class="d3-tip" style="position: absolute; top: 0px; opacity: 0;
      pointer-events: none; box-sizing: border-box;"></div>
    </body>
</html>
```

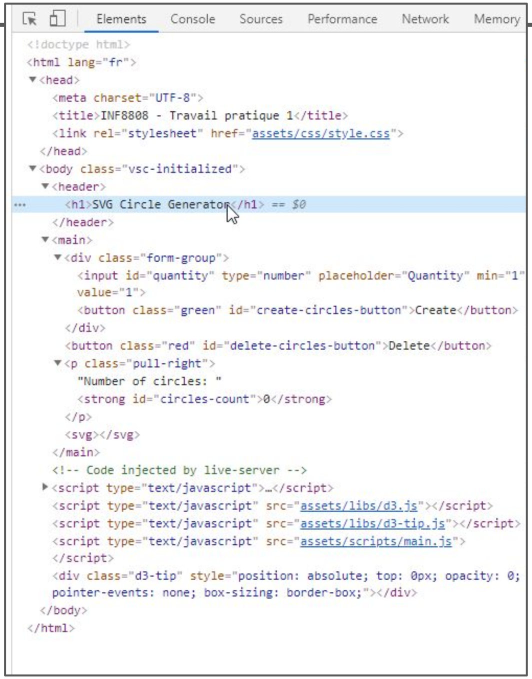
```
element.style {
}
html[Attributes Style] {
   -webkit-locale: "fr";
}
html {                          user agent stylesheet
   display: block;
}
```

Debugging with Chrome

Element inspection tool

- Shows the HTML structure (DOM) of your code
- Highlights the currently hovered element on the page
- Use it to verify your D3 code is correctly generating HTML elements
- Allows you to directly test changes to HTML and CSS
- For more info : [link]





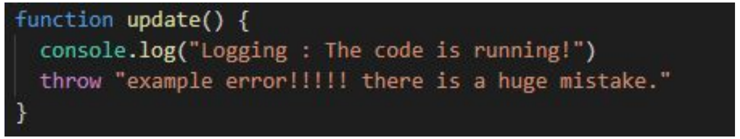
```
<!doctype html>
<html lang="fr">
 ▼<head>
    <meta charset="UTF-8">
    <title>INF8808 - Travail pratique 1</title>
    <link rel="stylesheet" href="assets/css/style.css">
  </head>
 ▼<body class="vsc-initialized">
   ▼<header>
...    <h1>SVG Circle Generator</h1> == $0
    </header>
   ▼<main>
     ▼<div class="form-group">
        <input id="quantity" type="number" placeholder="Quantity" min="1"
        value="1">
        <button class="green" id="create-circles-button">Create</button>
      </div>
      <button class="red" id="delete-circles-button">Delete</button>
     ▼<p class="pull-right">
        "Number of circles: "
        <strong id="circles-count">0</strong>
      </p>
      <svg></svg>
    </main>
    <!-- Code injected by live-server -->
   ▶<script type="text/javascript">…</script>
    <script type="text/javascript" src="assets/libs/d3.js"></script>
    <script type="text/javascript" src="assets/libs/d3-tip.js"></script>
    <script type="text/javascript" src="assets/scripts/main.js">
    </script>
    <div class="d3-tip" style="position: absolute; top: 0px; opacity: 0;
    pointer-events: none; box-sizing: border-box;"></div>
  </body>
</html>
```

Inspecting an h1 element

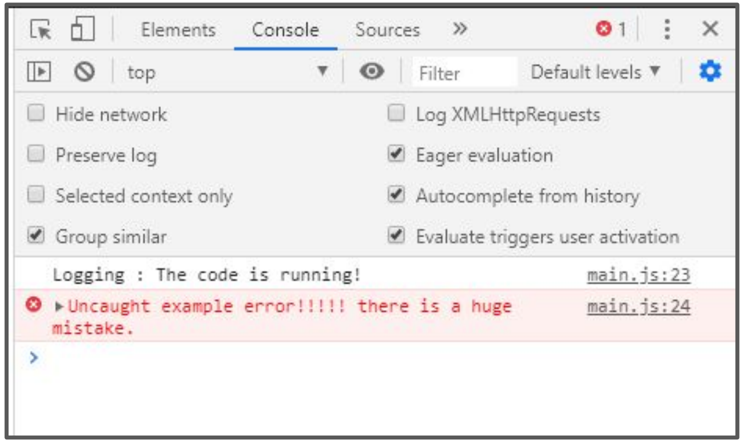
Debugging with Chrome

Console tool

- In the console, you will see outputs from your code
- These may include error messages and logs
- If something is not working, this is the first place you should look
- For more info : [link]



```
function update() {
  console.log("Logging : The code is running!")
  throw "example error!!!!! there is a huge mistake."
}
```



☐ Hide network
☐ Preserve log
☐ Selected context only
☑ Group similar
☐ Log XMLHttpRequests
☑ Eager evaluation
☑ Autocomplete from history
☑ Evaluate triggers user activation

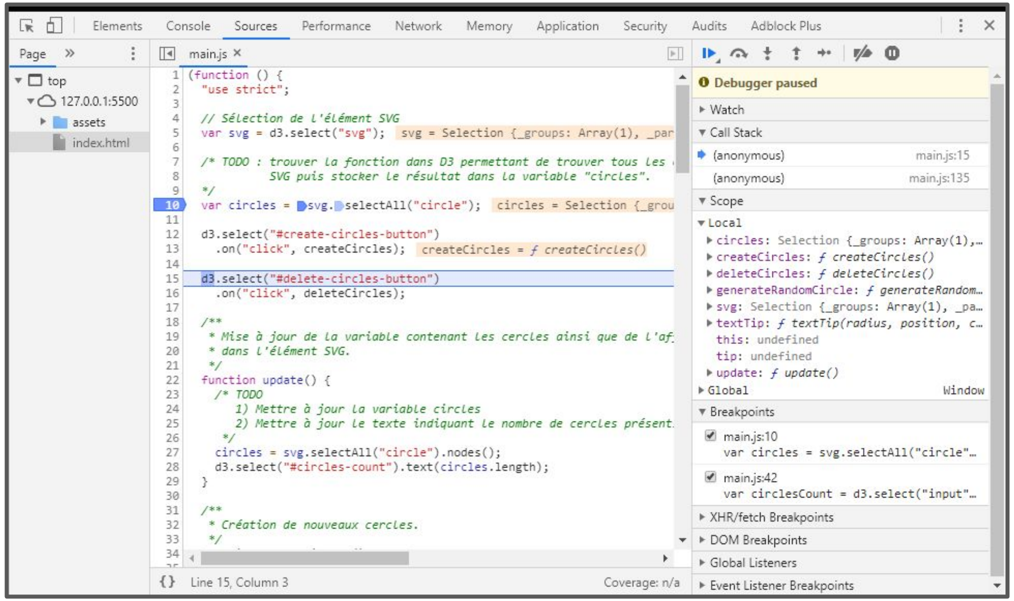
```
Logging : The code is running!                         main.js:23
▶Uncaught example error!!!!! there is a huge          main.js:24
mistake.
>
```

Logs and errors appear in the console

Debugging with Chrome

Sources inspection tool

- In this tab, you can see your source code and test modification to it
- You can also add breakpoints, where the execution will stop
- From a breakpoint, you can see the value of each variable and step through the code line by line
- For more info : [link]



```
(function () {
  "use strict";

  // Sélection de l'élément SVG
  var svg = d3.select("svg");  svg = Selection {_groups: Array(1), _par

  /* TODO : trouver la fonction dans D3 permettant de trouver tous les
            SVG puis stocker le résultat dans la variable "circles".
  */
  var circles = svg.selectAll("circle");  circles = Selection {_grou

  d3.select("#create-circles-button")
    .on("click", createCircles);  createCircles = ƒ createCircles()

  d3.select("#delete-circles-button")
    .on("click", deleteCircles);

  /**
   * Mise à jour de la variable contenant les cercles ainsi que de l'af
   * dans l'élément SVG.
   */
  function update() {
    /* TODO
       1) Mettre à jour la variable circles
       2) Mettre à jour le texte indiquant le nombre de cercles présent.
     */
    circles = svg.selectAll("circle").nodes();
    d3.select("#circles-count").text(circles.length);
  }

  /**
   * Création de nouveaux cercles.
   */
```

```
Line 15, Column 3                                    Coverage: n/a
```

```
Debugger paused
▸ Watch
▾ Call Stack
  (anonymous)                                   main.js:15
  (anonymous)                                   main.js:135
▾ Scope
  ▾ Local
    ▸ circles: Selection {_groups: Array(1),…
    ▸ createCircles: ƒ createCircles()
    ▸ deleteCircles: ƒ deleteCircles()
    ▸ generateRandomCircle: ƒ generateRandom…
    ▸ svg: Selection {_groups: Array(1), _pa…
    ▸ textTip: ƒ textTip(radius, position, c…
      this: undefined
      tip: undefined
    ▸ update: ƒ update()
  ▸ Global                                      Window
▾ Breakpoints
  ☑ main.js:10
    var circles = svg.selectAll("circle"…
  ☑ main.js:42
    var circlesCount = d3.select("input"…
▸ XHR/fetch Breakpoints
▸ DOM Breakpoints
▸ Global Listeners
▸ Event Listener Breakpoints
```

Parcours de code

Web technologies

Technologies

From the course outline :

“Following the completion of the lab works, the student will be able to develop a web interactive visualization ***using the D3.js library****, from a dataset and their conception work.”*

Thus, for the labs we will use :

- HTML / SVG / CSS
- JavaScript + D3.js

Often, the TPs can be accomplished many different ways - we expect you to choose the option that uses these technologies!

Technologies

HTML

Hypertext Markup Language is used to structure content for web browsers

```
<!DOCTYPE html>
<html>
        <head>
                <title>Page Title</title>
        </head>
        <body>
                <h1>Page Title</h1>
                <p>This is a really interesting paragraph.</p>
        </body>
</html>
```

- <...> open, </…> close
- DOM - Document Object Model (hierarchical structure of HTML)
  - <...> </…> is an element
  - Relationships:
    - Siblings (h1 and p are siblings)
    - Childrens (h1 and p)
    - Parent (body)
    - Descendant (html)

Technologies

SVG

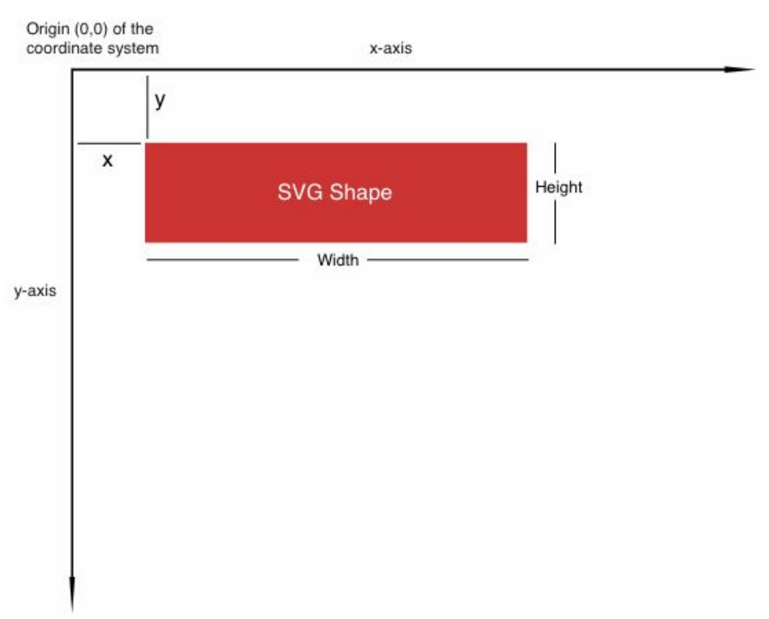
- “Scalable Vector Graphic”
- Useful HTML element for data visualisations

Examples :

- <circle>
  - Attributes : cx, cy, r
- <rectangle>
  - Attributes : x, y, width, height

Be careful! The coordinates system starts in the upper left hand corner

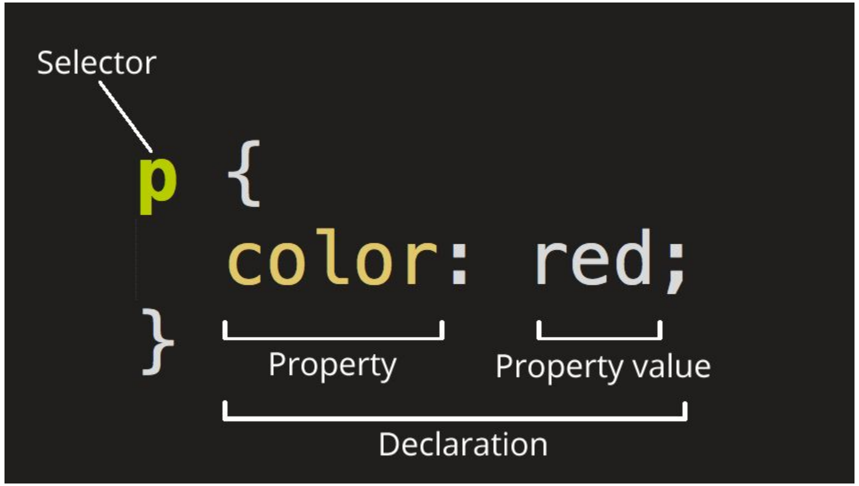
Rectangle SVG shape and coordinates system (source)



Technologies

CSS

- Describes the presentation of an HTML page
- Uses **selectors** and **declarations**



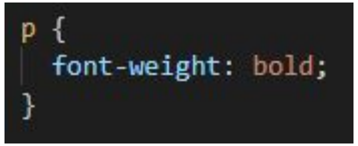
```css
p {
  color: red;
}
```

CSS (source)

Technologies

CSS selectors and declarations

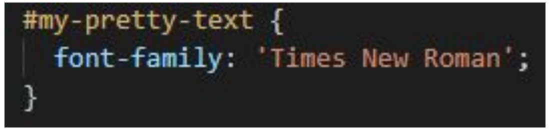
- Examples of selectors
  - By type of element



```
p {
  font-weight: bold;
}
```

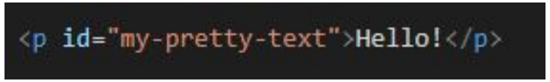
All <p> elements will be bold

  - On a single element : **by id**



```
#my-pretty-text {
  font-family: 'Times New Roman';
}
```

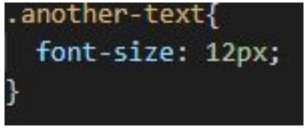
Font of element with id "my-pretty-text" will be "Times New Roman"



```
<p id="my-pretty-text">Hello!</p>
```

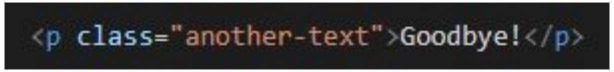
Corresponding HTML element

  - On a group of elements : **by class**



```
.another-text{
  font-size: 12px;
}
```

Font of elements with class "another-text" will be 12px



```
<p class="another-text">Goodbye!</p>
```

Corresponding HTML element

Technologies

CSS selectors and declarations

- Declarations
  - Depend on the style attributes of the selected element(s)

- Examples…

- **width and height** : width and height of an element
- **fill** : color of an element
- **margin** : margin around an element
- **font-size** : size of the font of text element

Technologies

JavaScript

Scripting language that can make pages dynamic by manipulating the DOM.

It is interpreted & weakly typed.

Points to highlight:

- Objects and arrays: know how to combine these two structures help to process the data.
- JSON: specific syntax for organizing data as JavaScript objects. Easier to parse and better for D3 works.
- Functions: take arguments or parameters as input, and then return values as output.

Technologies

JavaScript

Referencing Scripts:

- Directly in HTML, between two *scripts* tags:

```
<body>
    <script type="text/javascript">
        alert("Hello, world!");
    </script>
</body>
```

- Stored in a separate file with a .js suffix, and then referenced somewhere in HTML:

```
<head>
    <title>Page Title</title>
    <script type="text/javascript" src="myscript.js"></script>
</head>
```

Most used on TP's

Technologies

D3.js

- D3 stands for “Data-driven document”
- JS library
- We will use version 5 (**Important**: When searching for examples make sure they are > v4, or else they might be quite different!)
- D3 can be used to process and visualize data, ex :
    - d3.select(‘svg’).append(‘circle’).attr(‘r’,10);
    - Selects the first element with tag “svg” and appends to it a circle of radius 10
- We will see D3 in more detail in the readings and exercises
- We will also see more D3 in the next labs!

Guidelines

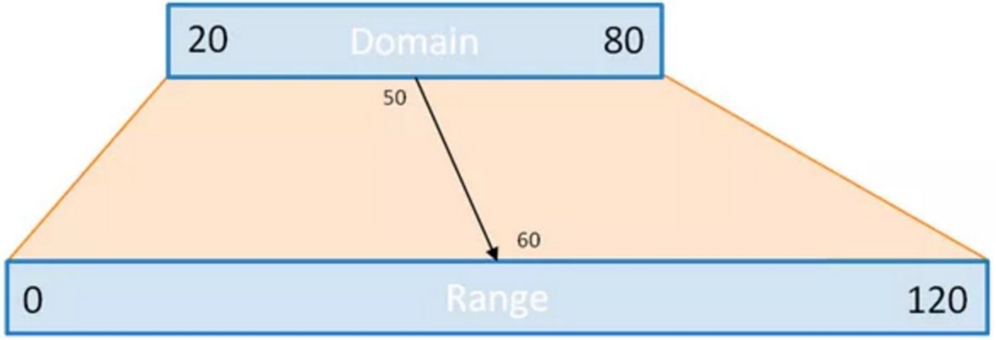
D3 Scales

- In **viz.js**, you need to determine the domain and range of scales in D3

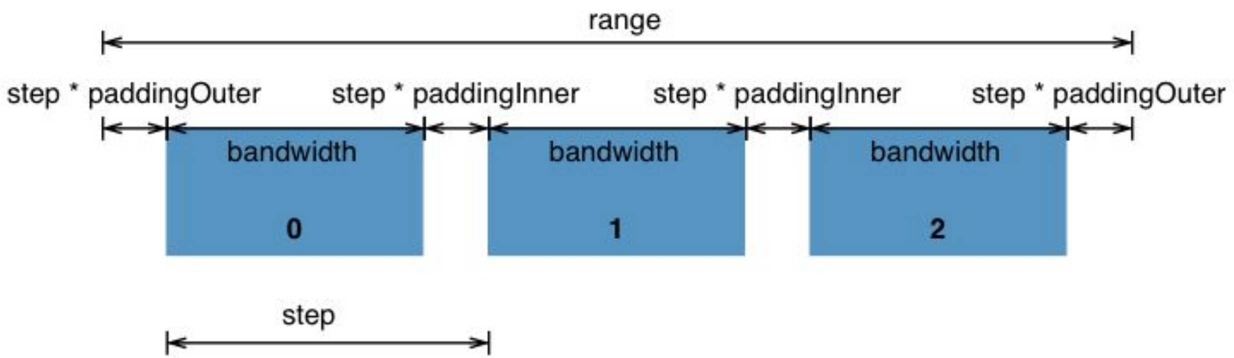
- In this TP, the scales help draw the axes and position the groups of rectangles, as well as the rectangles within the groups
    - Scales : Color scale, X scale (per group), X scale (for each rectangle within the group), Y scale for length of rectangles

D3 scales

Visually



Source : *https://medium.com/@sahilaug/line-graphs-using-d3-drawing-the-axes-8ffc0076a8be*



Source : *https://github.com/d3/d3-scale*

Guidelines

Data Binding

- In this TP use **data binding** features of D3
- 2 structures :

1. Classic :

```
d3
 .selectAll('rect')
 .data(myData)
 .enter()
 .append('rect')
```

2. Newer:

```
d3
 .selectAll('rect')
 .data(myData)
 .join('rect')
```

- Reference : https://github.com/d3/d3-selection

Data binding in D3

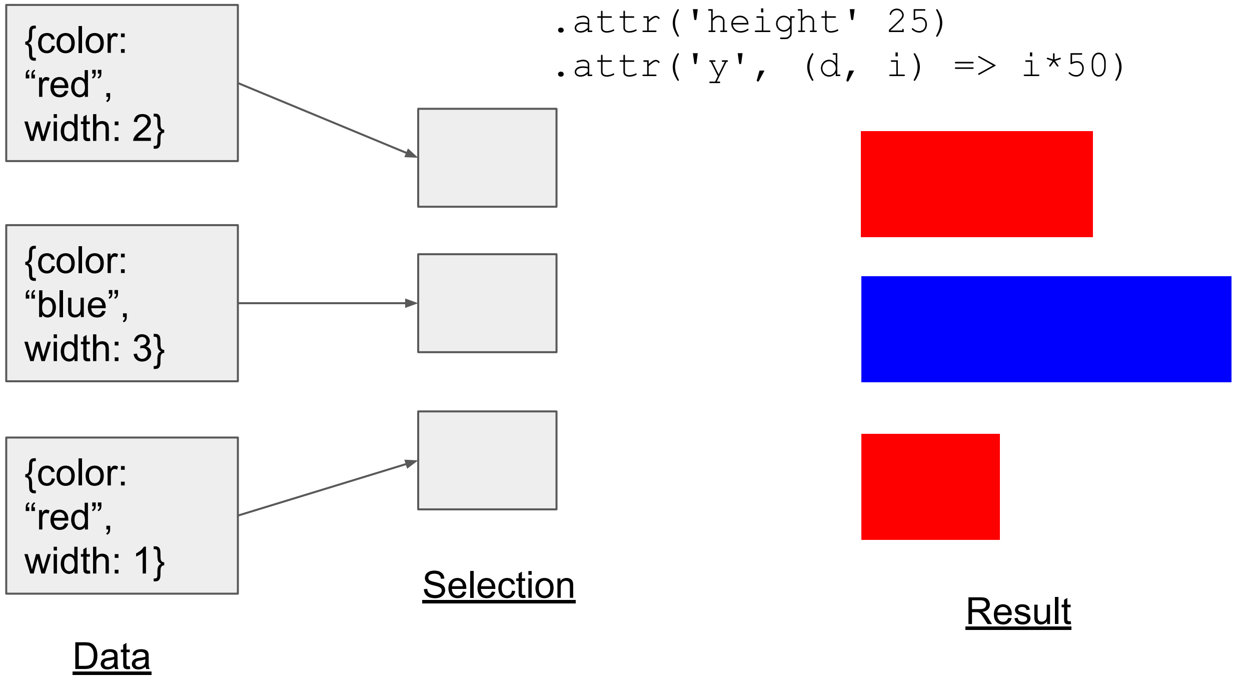
Visually

rectangles.js

```
svg.selectAll('rect')
.data(data)
.enter()
.append('rect')
.attr('fill', (d) => d.color)
.attr('width', (d) => d.width)
.attr('height' 25)
.attr('y', (d, i) => i*50)
```

data.csv

```
color, width
red,   2
blue,  3
red,   1
```



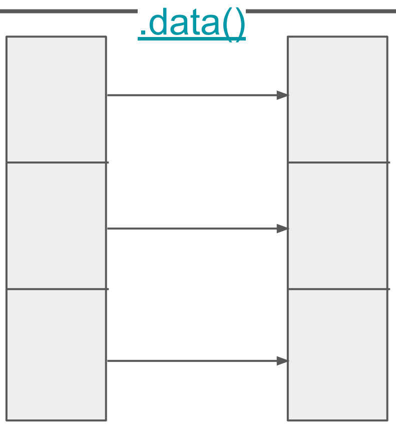
Data binding in D3

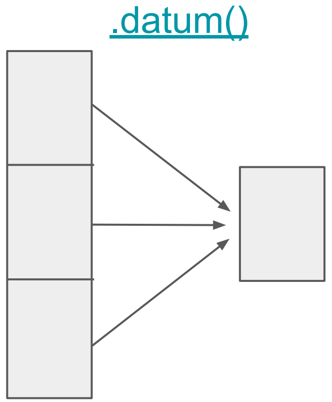
.data() VS .datum()

Two functions with similar names but different use cases

- **.data()** binds one data point per element
- **.datum()** bind all data points to each element

***Hint :* Usually, the TPs require .data()**

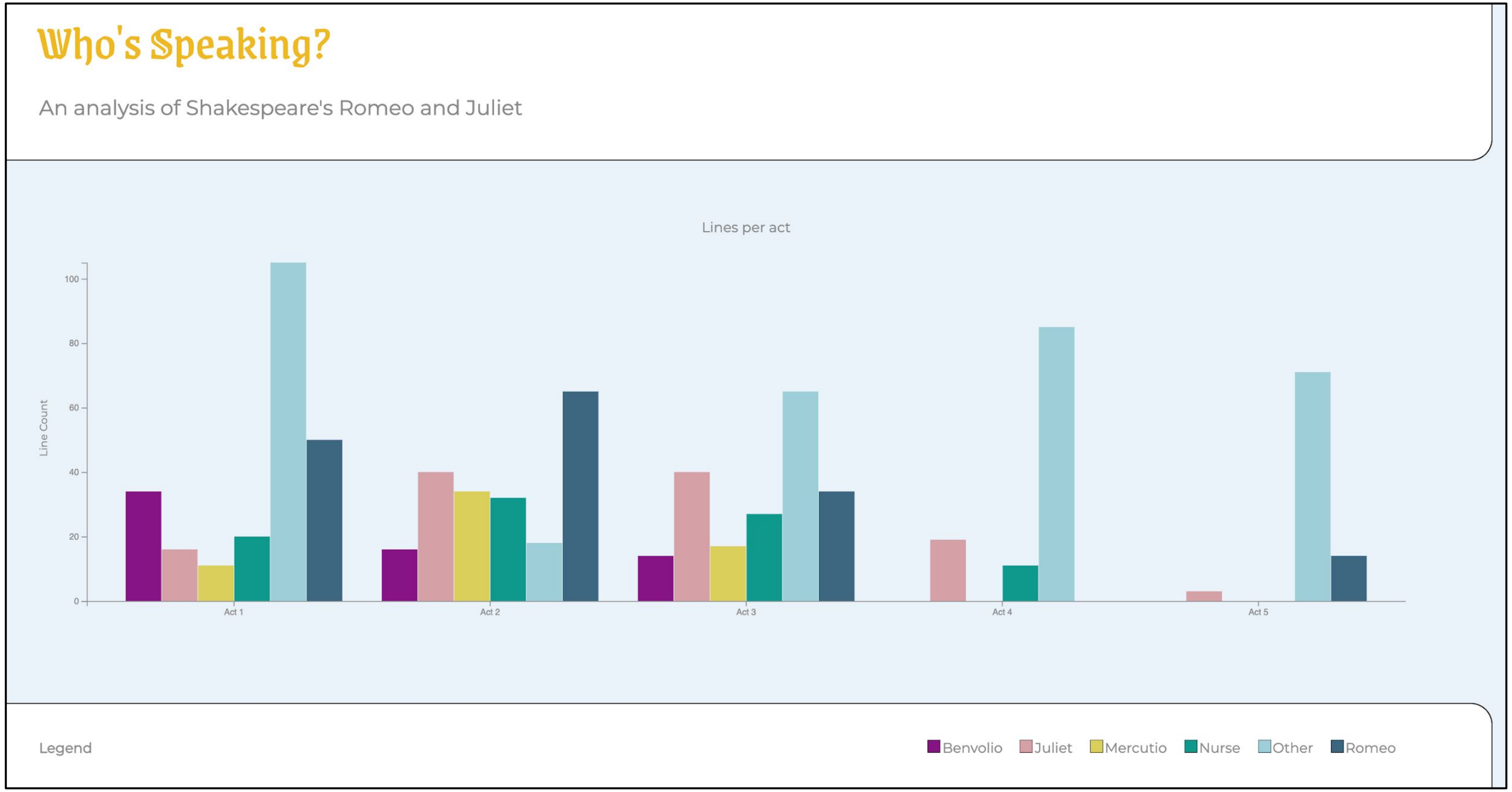




TP2

Introduction to TP2

- Bar chart
- Data from *Romeo and Juliet*
- Result:



Who's Speaking?

An analysis of Shakespeare's Romeo and Juliet

To run the code

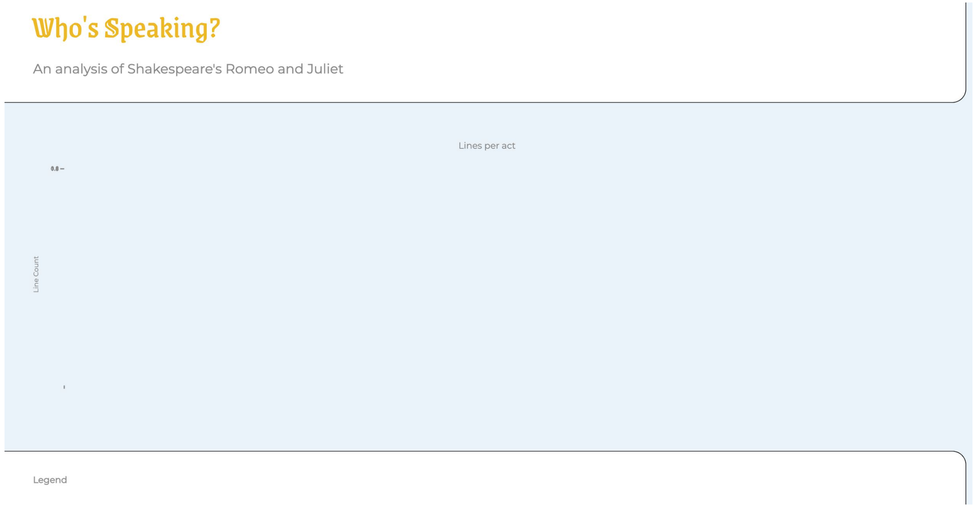
- In a terminal, at the same level as *package.json:*

```
npm install

npm start
```

- Then see localhost:8080 in your browser

Result:

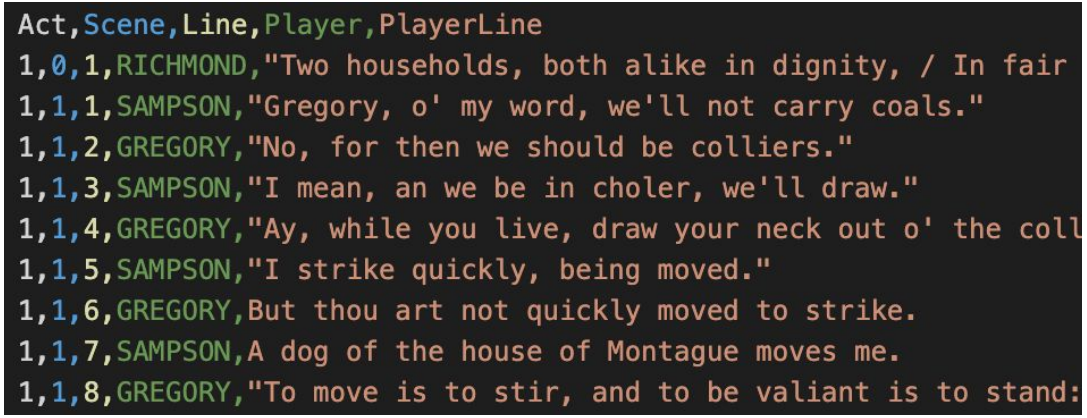

Who's Speaking?

An analysis of Shakespeare's Romeo and Juliet

Dataset

The dataset is located in the `src/assets/data/` directory in the archive provided for the lab. The dataset contains the following columns :

- **Act**
- **Scene**
- **Line**
- **Player**
- **PlayerLine**

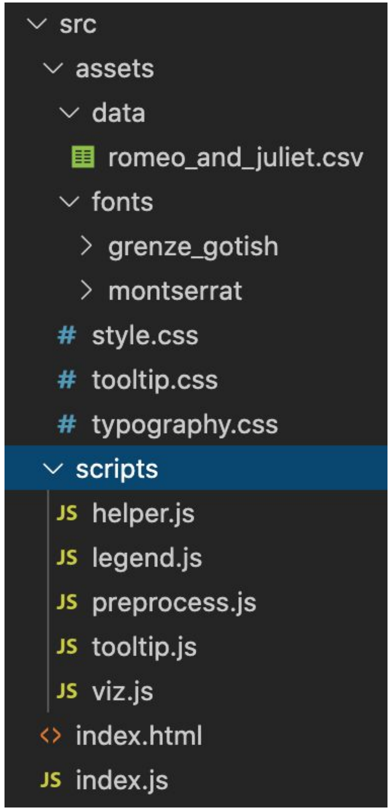


```
Act,Scene,Line,Player,PlayerLine
1,0,1,RICHMOND,"Two households, both alike in dignity, / In fair
1,1,1,SAMPSON,"Gregory, o' my word, we'll not carry coals."
1,1,2,GREGORY,"No, for then we should be colliers."
1,1,3,SAMPSON,"I mean, an we be in choler, we'll draw."
1,1,4,GREGORY,"Ay, while you live, draw your neck out o' the coll
1,1,5,SAMPSON,"I strike quickly, being moved."
1,1,6,GREGORY,But thou art not quickly moved to strike.
1,1,7,SAMPSON,A dog of the house of Montague moves me.
1,1,8,GREGORY,"To move is to stir, and to be valiant is to stand:
```

Tasks :

1. **Process data**
   - File : ./src/scripts/*preprocess.js*
2. **Make bar chart**
   - File : ./src/scripts/*viz.js*
3. **Make legend**
   - File : ./src/scripts/*legend.js*
4. **Add tooltip**
   - File : ./src/scripts/*tooltip.js*

DO NOT MODIFY OTHER FILES.



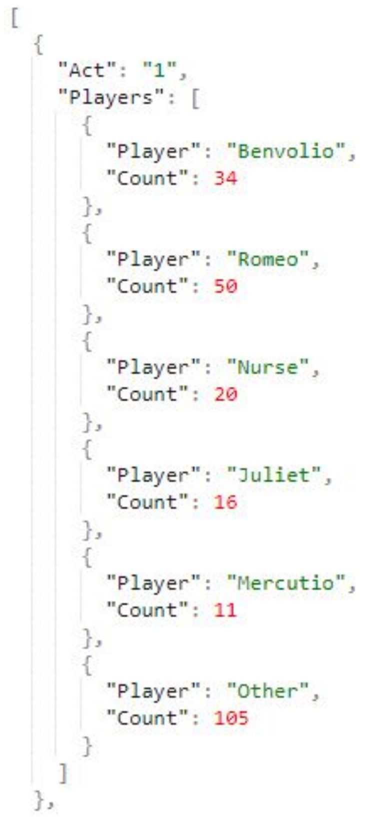
```
src
  assets
    data
      romeo_and_juliet.csv
    fonts
      grenze_gotish
      montserrat
    style.css
    tooltip.css
    typography.css
  scripts
    helper.js
    legend.js
    preprocess.js
    tooltip.js
    viz.js
  index.html
  index.js
```

1. Preprocess data

4 functions to fill in the file **preprocess.js** :

1. cleanName
2. getTopPlayers
3. summarizeLines
4. replaceOthers

The result should look like the structure on the right :



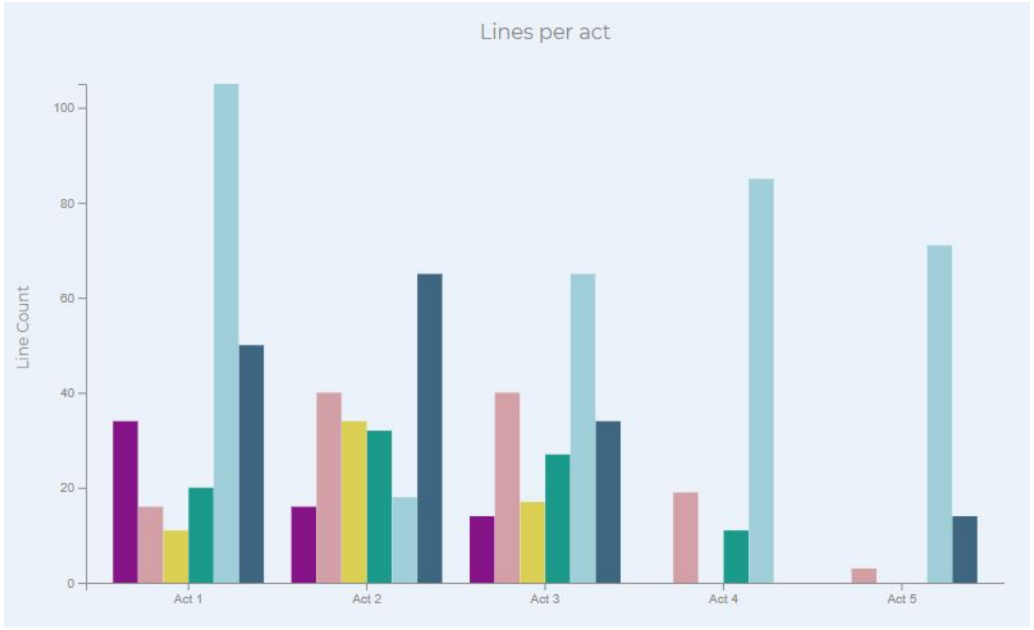
```
[
  {
    "Act": "1",
    "Players": [
      {
        "Player": "Benvolio",
        "Count": 34
      },
      {
        "Player": "Romeo",
        "Count": 50
      },
      {
        "Player": "Nurse",
        "Count": 20
      },
      {
        "Player": "Juliet",
        "Count": 16
      },
      {
        "Player": "Mercutio",
        "Count": 11
      },
      {
        "Player": "Other",
        "Count": 105
      }
    ]
  },
```

2. Bar chart

4 functions in **viz.js** for the bar chart :

1. updateGroupScale
2. updateYScale
3. createGroups
4. drawBars

The result of this part is to the right :

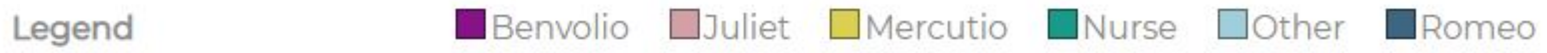


3. Legend

Fill the code in **legend.js** to trace the legend.

Pay special attention to the given HTML and CSS classes to help.

The result of this part is as follows :



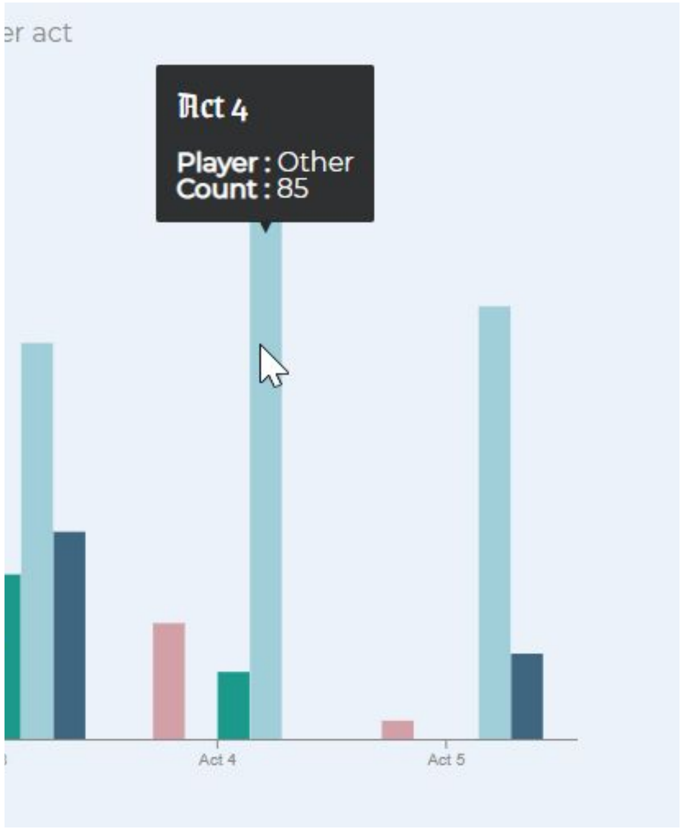
4. Tooltip :

Fill the code in **tooltip.js** for the tooltip.

The tooltip contains the act label, player name and line count.

Pay attention to make it look as shown in the TP subject.

The result is like here at the right :



Some advice to start

Data preprocessing

- Avoid directly manipulating data indices where possible (As in : index `i` → `data[i]`)
  - This will improve quality and maintainability of code, while reducing risk of errors
- Instead explore methods such as :
  - `.foreach()`
  - `.map()`
  - `.reduce()`
  - `.filter()`
  - `.find()`
  - And in D3, make sure to use `.enter()`
- Using `for ... of` is also sometimes useful

Quality and clarity of submission

More details

Each TP will also be graded on **overall quality and clarity of the submission**

Examples

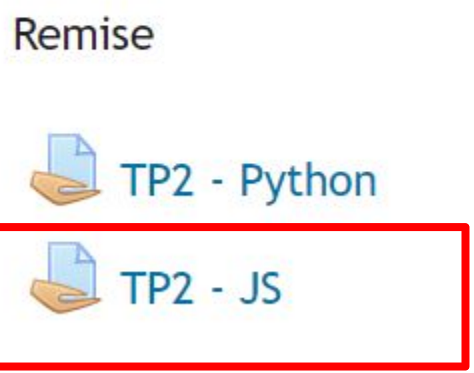
- Clear code structure
- Do not modify signatures of existing functions
- You can add new functions, but they must be clear and their addition must be justified
- Use clear indentations
- Add comments as needed, but not too many
- Don’t leave dead code
- Don’t leave useless console.log
- Make sure to follow instructions for submission

Etc.

SUBMISSION GUIDELINES

- One submission per team

- Submit in the JS submission box on Moodle


Remise

TP2 - Python

TP2 - JS

SUBMISSION GUIDELINES

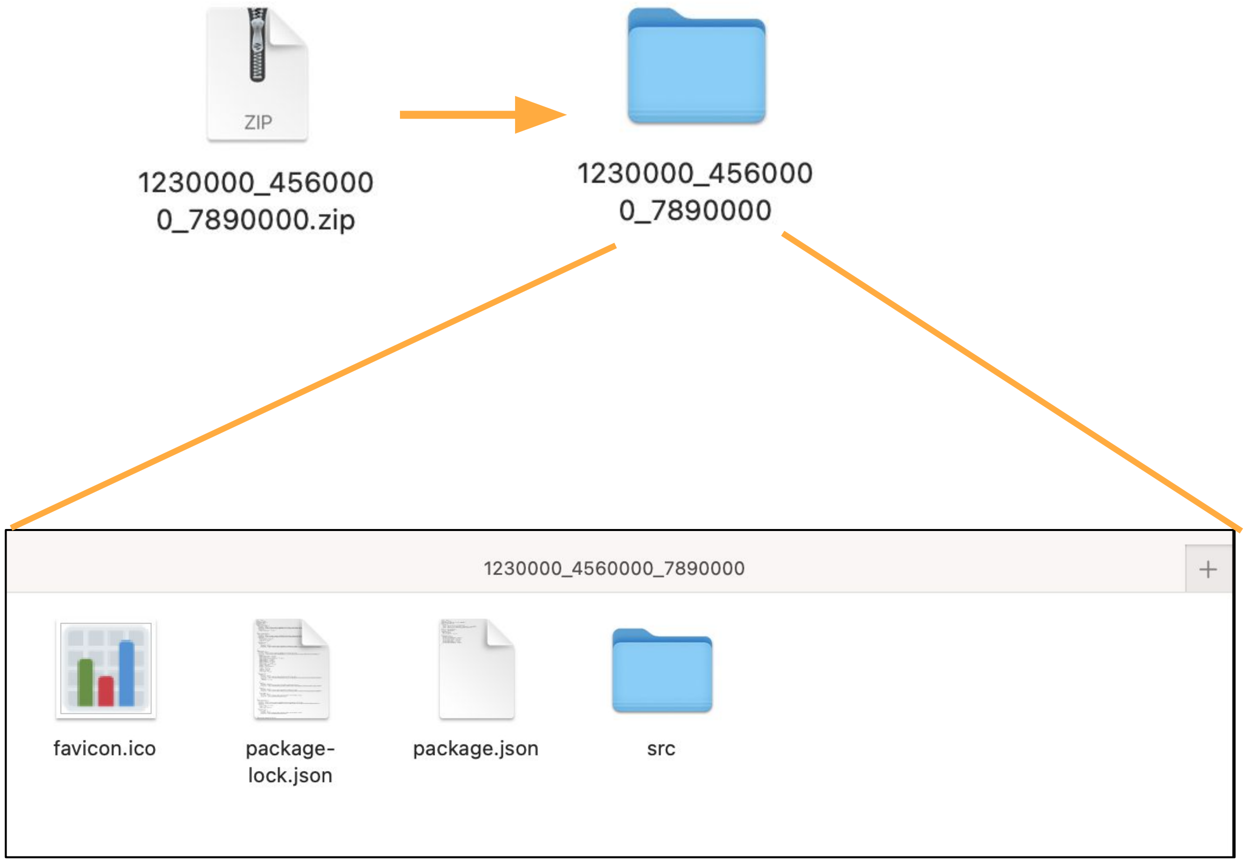
- On Moodle, submit a .zip named **studentid1_studentid2_studentid3_.zip**

- The zip must contain the same files and structure as when initially downloaded from moodle

- Do **not** include node_modules, .cache and dist folders

- Zip just the initial files, do not create any new folders in the .zip

Result example:


ZIP
1230000_4560000_7890000.zip
1230000_4560000_7890000
1230000_4560000_7890000
favicon.ico
package-lock.json
package.json
src